

S 59  
IC  
6  
a coprocessor for performing [vector type]  
operations on a plurality of components of one pixel of the  
graphics data; and

7  
8 a direct memory access (DMA) engine for transferring  
9 the graphics data between an external memory and the local memory.

1 (Unchanged) 2. The graphics accelerator of claim 1 wherein the  
2 memory includes a data SRAM.

1 (Canceled) 3. The graphics accelerator of claim 1 further  
2 comprising a direct memory access (DMA) engine for loading the  
3 graphics data from memory through loading operations and  
4 transferring processed graphics data to the memory through storing  
5 operations.

1 (Unchanged) 4. The graphics accelerator of claim 1 wherein the  
2 coprocessor processes the plurality of components of each pixel in  
3 parallel as three elements of a vector.

1 (Unchanged) 5. The graphics accelerator of claim 4 wherein the  
2 plurality of components of each pixel comprise R, G and B  
3 components of RGB formatted graphics data.

1 (Unchanged) 6. The graphics accelerator of claim 5 wherein the  
2 pixels are in an RGB16 format.

1 (Unchanged) 7. The graphics accelerator of claim 6 wherein the  
2 R component has five bits, the G component has six bits and the B  
3 component has 5 bits.

1 (Unchanged) 8. The graphics accelerator of claim 6 wherein the  
2 graphics data is organized into 32-bit words, and each 32-bit word  
3 includes two pixels having RGB16 format.

1 (Unchanged) 9. The graphics accelerator of claim 8 wherein the  
2 two pixels are respectively selected by two special load  
3 instructions.

1 (Unchanged) 10. The graphics accelerator of claim 9 wherein the  
2 two special load instructions are for loading a left one and a  
3 right one of the two pixels, respectively.

1 (Unchanged) 11. The graphics accelerator of claim 7 wherein the  
2 coprocessor comprises an input register.

1 (Unchanged) 12. The graphics accelerator of claim 11 wherein  
2 the R, G and B components are expanded into 8-bit components  
3 through zero expansion when loaded into the input register.

1 (Unchanged) 13. The graphics accelerator of claim 4 wherein the  
2 plurality of components of each pixel comprise Y, U and V  
3 components of YUV formatted graphics data, and  
4 wherein the Y, U and V components are also referred to as Y,  
5 Cr and Cb components, respectively, of YCrCb formatted graphics  
6 data.

1 (Unchanged) 14. The graphics accelerator of claim 13 wherein  
2 the pixels are in a YUV 4:2:2 format.

1 (Unchanged) 15. The graphics accelerator of claim 14 wherein  
2 the pixels are organized into 32-bit words and each 32-bit word  
3 contains two pixels.

1 (Unchanged) 16. The graphics accelerator of claim 15 wherein  
2 the two pixels in each 32-bit word is organized in a YUYV format,  
3 each of the first Y component, the U component, the second Y

4 component, and the V component occupies eight bits, a first one of  
5 the two pixels is comprised of a first Y component, the U component  
6 and the V component, and a second one of the two pixels is  
7 comprised of the second Y component, the U component and the V  
8 component.

1 (Unchanged) 17. The graphics accelerator of claim 15 wherein  
2 the two pixels are respectively selected by two special load  
3 instructions.

1 (Unchanged) 18. The graphics accelerator of claim 17 wherein  
2 the two special load instructions are for extracting a first one  
3 and a second one of the two pixels, respectively.

1 (Unchanged) 19. The graphics accelerator of claim 4 wherein the  
2 coprocessor has an instruction set that includes a special  
3 instruction for comparing between each element of a pair of 3-  
4 element vectors.

1 (Unchanged) 20. The graphics accelerator of claim 19 wherein  
2 the coprocessor further comprises a result register, and results of  
3 the three comparisons are stored in the result register.

1 (Unchanged) 21. The graphics accelerator of claim 20 wherein  
2 the results of the three comparisons are used together during a  
3 single conditional branch operation.

1 (Unchanged) 22. The graphics accelerator of claim 19 wherein  
2 the special instruction is for a greater-than-or-equal-to  
3 operation.

1 (Amended) 23. The graphics accelerator of claim 3 wherein the  
2 DMA engine moves data between the local memory and [an] the  
3

448  
C43  
Cml  
external memory at the same time the graphics accelerator is using the memory for its load and store operations.

---

1 (Unchanged) 24. The graphics accelerator of claim 23 wherein  
2 the external memory is a unified memory that is shared by a  
3 graphics display system, a CPU and other peripheral devices.

1 (Unchanged) 25. The graphics accelerator of claim 3 wherein the  
2 DMA engine includes a queue to hold a plurality of DMA commands.

1 (Unchanged) 26. The graphics accelerator of claim 25 wherein  
2 the plurality of DMA commands are executed in the order they are  
3 received.

1 (Unchanged) 27. The graphics accelerator of claim 25 wherein  
2 the queue comprises a mechanism that allows the graphics  
3 accelerator to determine when all the DMA commands have been  
4 completed.

1 (Unchanged) 28. The graphics accelerator of claim 25 wherein  
2 the queue is four deep for storing up to four DMA commands.

1 (Unchanged) 29. The graphics accelerator of claim 3 wherein the  
2 graphics accelerator is working on operands and producing outputs  
3 for one set of pixels, while the DMA engine is bringing in operands  
4 for a future set of pixel operations.

1 (Amended) 30. A method of processing graphics comprising the  
steps of:

4 loading a block of graphics data from main memory  
into local memory of a graphics accelerator having a processor and  
5 a coprocessor, the graphics data including pixels, each pixel  
6 having a plurality of components; [and]

7 performing [vector type] operations on the plurality  
8 of components of each pixel of graphics data using the coprocessor;  
9 and

10 concurrently transferring blocks of unprocessed data  
11 and processed data between the main memory and the local memory  
12 while the block of graphics data is being processed.

1 (Unchanged) 31. The method of processing graphics of claim 30  
2 wherein the plurality of components comprises R, G and B components  
3 of RGB formatted graphics data.

1 (Unchanged) 32. The method of processing graphics of claim 31  
2 wherein the pixels of the graphics data are in an RGB16 format.

1 (Unchanged) 33. The method of processing graphics of claim 32  
2 further comprising the step of organizing the graphics data into  
3 32-bit words, wherein each 32-bit word includes two pixels having  
4 RGB16 format.

1 (Unchanged) 34. The method of processing graphics of claim 33  
2 further comprising the step of selecting each of the two pixels  
3 with one of two special load instructions.

1 (Unchanged) 35. The method of processing graphics of claim 34  
2 wherein the step of selecting each of the two pixels comprises  
3 loading a left one of the two pixels.

1 (Unchanged) 36. The method of processing graphics of claim 34  
2 wherein the step of selecting each of the two pixels comprises  
3 loading a right one of the two pixels.

1 (Unchanged) 37. The method of processing graphics of claim 30  
2 wherein each of the plurality of pixels of graphics data comprises  
3 Y, U and V components of YUV formatted graphics data.

1 (Unchanged) 38. The method of processing graphics of claim 37  
2 wherein the pixels are in a YUV 4:2:2 format.

1 (Unchanged) 39. The method of processing graphics of claim 38  
2 further comprising the step of organizing the pixels into 32-bit  
3 words, wherein each 32-bit word contains two pixels.

1 (Unchanged) 40. The method of processing graphics of claim 39  
2 wherein the step of organizing the pixels into 32-bit words  
3 comprises organizing each of the two pixels into a YUYV format,  
4 wherein each of the first Y component, the U component, the second  
5 Y component and the V component occupies eight bits, a first one of  
6 the two pixels is comprised of a first Y component, the U component  
7 and the V component, and a second one of the two pixels is  
8 comprised of the second Y component, the U component and the V  
9 component.

1 (Unchanged) 41. The method of processing graphics of claim 40  
2 further comprising the step of selecting each of the two pixels  
3 with one of two special load instructions.

1 (Unchanged) 42. The method of processing graphics of claim 41  
2 wherein the step of selecting each of the two pixels comprises  
3 loading the first one of the two pixels.

1 (Unchanged) 43. The method of processing graphics of claim 42  
2 wherein the step of selecting each of the two pixels comprises  
3 loading the second one of the two pixels.

1 (Unchanged) 44. The method of processing graphics of claim 30  
2 further comprising the step of comparing between each element of a  
3 pair of 3-element vectors, wherein each element of the 3-element  
4 vector is one of three components of each pixel.

1 (Unchanged) 45. The method of processing graphics of claim 44  
2 wherein the three components of each pixel are R, G and B  
3 components.

1 (Unchanged) 46. The method of processing graphics of claim 44  
2 wherein the three components of each pixel are Y, U and V  
3 components.

1 (Unchanged) 47. The method of processing graphics of claim 44  
2 wherein the coprocessor includes a result register, and the method  
3 further comprising the step of storing results of the three  
4 comparisons in the result register.

1 (Unchanged) 48. The method of processing graphics of claim 47  
2 further comprising the step of performing a single conditional  
3 branch operation using the results of the three comparisons.

1 (Unchanged) 49. The method of processing graphics of claim 44  
2 wherein the step of comparing between each element of a pair of 3-  
3 element vectors comprises the step of performing a greater-than-or-  
4 equal-to operation between each element of a pair of 3-element  
5 vectors.

24/ 50. The method of processing graphics of claim 30 4/  
1 (Amended) 24/ 50. The method of processing graphics of claim 30 4/  
2 wherein the graphics accelerator includes [a] the local memory for  
3 loading the graphics data, the method further comprising the step  
4 of moving data between the local memory and [an] the external  
5 memory using a direct memory access (DMA) engine at the same time  
6 the graphics accelerator is using the local memory for its load and  
7 store operations.

25/ 51. The method of processing graphics of claim 50 24/  
1 (Amended) 25/ 51. The method of processing graphics of claim 50 24/  
2 wherein the local memory is a data SRAM.

1 (Unchanged) 52. The method of processing graphics of claim 50  
2 wherein the external memory is a unified memory that is shared by  
3 a graphics display system, a CPU and other peripheral devices.

1 (Unchanged) 53. The method of processing graphics of claim 50  
2 wherein the DMA engine includes a queue for holding a plurality of  
3 DMA commands.

1 (Unchanged) 54. The method of processing graphics of claim 53  
2 further comprising the step of determining whether all the DMA  
3 commands have been completed.

1 (Unchanged) 55. The method of processing graphics of claim 53  
2 further comprising the step of receiving the plurality of DMA  
3 commands.

1 (Unchanged) 56. The method of processing graphics of claim 55  
2 further comprising the step of executing the plurality of DMA  
3 commands in the order they are received.

1 (Unchanged) 57. The method of processing graphics of claim 53  
2 wherein the queue is four deep for storing up to four DMA commands.

1 (Unchanged) 58. The method of processing graphics of claim 30  
2 further comprising the step of bringing in operands for a future  
3 set of pixel operations using a direct memory access (DMA) engine  
4 while the graphics accelerator is working on operands and producing  
5 outputs for one set of pixels.

[Please add new claims 59-61 as follows:]

SUB  
CO  
1 (New) -- 59. The graphics accelerator of claim 1, wherein the DMA  
2 includes a queue for storing DMA commands, wherein the processor  
3 posts a plurality of commands to the queue and the DMA executes the  
4 commands concurrently with the operation of the processor. --

B  
1 (New) -- 60. The graphics accelerator of claim 1 wherein the  
2 coprocessor incorporates load and store functions for unpacking  
3 graphics data into a plurality of components, processing each of  
4 the plurality of components, and converting the plurality of  
5 components into a format suitable for storage. -

SUB  
CO  
1 (New) -- 61. A method for sequentially processing blocks of  
2 graphics data, the method comprising the steps of:  
3 transferring a first block of unprocessed graphics  
4 data from main memory to on-board memory;  
5 processing the first block of graphics data;  
6 transferring a second block of unprocessed graphics  
7 data from the main memory to the on-board memory while the first  
8 block of graphics data is being processed; and  
9 transferring a third block of processed graphics  
10 data from the on-board memory to the main memory while the first  
11 block of graphics data is being processed. --